International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 3, Issue 5, Dec 2013, 27-42

CASE STUDY: SCALING AGILE METHODS USING RETROSPECTIVE REVIEWS PROCESS

SARAVANA K. M[1], G. N. BASAVARAJ[2], RAJKUMAR[3] & H G CHANDRAKANTH[4]

[1]Lead Engineer, GXS ITC Private Limited, Bangalore, Karnataka, India

[2,3,4]Assistant Professor, Principal, Department of ISE, Sambhram Institute of Technology, Bangalore, Karnataka, India

ABSTRACT

Knowledge from experience is the fundamental to victories for all that develop software. Together the achievements and the disappointments in agile software projects can assistance us to progress. Now we deliberate Agile Retrospective Reviews (ARR) is a modest and applied technique for sharing awareness succeeding the achievement of agile sprint iteration or major milestone projects. Nevertheless, not various companies have implemented such practices and in a review, limited communicated satisfaction through how retrospective were accompanied. The benefit is that retrospective can frequently reveal answers additional often and differently than project completion reports alone. Based on the answers, we present a lightweight technique. We deliberate that the approaches are valuable for companies when they essential to document their knowledge, discovery advance activities and as a start of systematic knowledge harvesting. We determine from our findings that ARR is worthwhile in small and medium-sized agile software development projects for which lightweight versions.

KEYWORDS: Project Analysis, Software Engineering, Agile Software Development, Retrospective Review, Process Improvement, Retrospective Method, Agile Development Methods

INTRODUCTION

Nowadays agile software engineering industry, it is critical towards evolution of software development practices. In this situation, unique moral that could be learned after overall efforts towards improve processes, such as overall quality management and standardization, is that the talent to learn from historical success and failure stands an essential aspect aimed at victory. Learning from the past involves knowledge management or creating a "learning software organization", which is defined by Dyba [1] as "A software organization that encourages enriched activities through healthier knowledge and understanding". Knowledge management has established greatly care in the software engineering arena during the past years, partially as a encouraging arena aimed at software process improvement in order towards growth quality and reduction budgets in software development. A shared aspect in knowledge management and in software process improvement is to acquire from historical achievements and failures in order to progress forthcoming software development. Experience Factory [2] has remained a vital term in aiming organizational wisdom on improving software development processes. Utmost companies that develop software establish the development in projects. In the Experience Factory, the projects are realized as the main arena for wisdom, and knowledge which seems in the projects is towards shared with other projects.

Agile software development approaches such as extreme programming have in the past years accomplished an explosive attention in the international information technology community. This can be realized as a reaction to the additional traditional and control-oriented approaches, where the waterfall technique and its variations are instances.

These approaches focus on certainty over wide-ranging scheduling and systematically survey on the strategies.

This approach of successively projects and develop software clashes through the increasing necessity to handle speedy variations in projects and their environment. It is now that agile approaches fit in. They are constructed to handle variations in design and requirements [3]. They exposed up for imagination throughout the entire project lifecycle and not fair in the initial planning and design stages. Agile means smooth, non-bureaucratic and adaptable.

The knowledge processes in agile methods, and specifically extreme programming as we will deliberate in this paper, are likewise agile. Those earnings that the knowledge process is basic associated to other additional complete development methodologies. Pair programming is unique of the extreme programming "practices" that are deliberate to support knowledge handover inside a team. This works healthy for pairs of programmers, but we privilege that learning can be even healthier if this is shared with additional learning modes. Agile promoters have distinguished that "each condition requests for a diverse methodology" [4, p. 184]. Thus, one of the principles behind agile manifesto recommends that the team should frequently replicate in what way to develop more effective, and fine-tune and regulate its performance consequently. Cockburn discusses to "the mystery of how to construct a diverse methodology for each condition without expenditure so much time designing the methodology" [4, p. 184]. Various systematic methodologies have been suggested in what way to achieve this self-reflection process successfully. Cockburn [4] recommends a methodology-growing technique including a team replication workshop after iteration. Furthermore, Hanssen [5] have recommended a learning mechanism called Retrospective reviews to be used as an extension for agile software development approaches. It works towards making worthy use of the experiences of project contributors at the end of iteration to improve the development process. In agile software development iteration might be from one to four weeks [6]. In terms of knowledge management, Retrospective reviews might be designated as a technique that targets "active collaboration that simplifies the transformation of personal knowledge into organizational knowledge" [7, p.14]. The knowledge of Retrospectives in software development projects is not a fresh one. In recent years diverse Retrospective techniques have been used in traditional software development approaches. They recommend that every project should determine with Retrospective review to evaluate our shortcomings in direction to acquire and progress [9]. Retrospective reviews have been established to be effective as a tool for organizational learning and productive from the software process improvement (SPI) point of view. Learning lessons from developments is significant, but the exercise of Retrospectives does not happen regularly in practice [10]. Recently it has been exposed that carefully scheduled and organized Retrospective analyses (i.e. not ad-hoc) can fetch unexpected benefits in practice.

Retrospective Analysis has considerably facilitated to progress and enhance practices in extreme programming. Popular another study by Stalhane et al. [11], unique key discovery was that "a software organization had not analyzed accurately, and through all the appropriate contributors, what was reusable from their previous projects". In their case Retrospective Analysis facilitated to recognize new enhancement likelihoods in the development process.

Retrospective Analysis is a simple and speedy approach of revealing information and knowledge around the numerous project characteristics. There are various diverse techniques of establishing project reviews. Some Retrospective Analysis implements and techniques are existing on the "Project Review" [12], and an explanation of the Retrospective process for minor projects can be establish in "Lightweight Retrospective Reviews" [13]. Welfares of applying both minor and huge Retrospectives, founded on the experiences described in [14,15]:

- Retrospectives benefits project team participants share and recognize each other's viewpoints;
- Retrospectives integrates individual and team learning;
- Retrospectives recognizes hidden complications;

- Retrospectives documents moral practices and complications (so as not to repeat immoral practices);
- Retrospectives increases job fulfillment by giving people feedback almost their effort;
- In certain cases, Retrospective Analysis can even improve project cost estimation.

The KJ-approach [8] through post-it transcripts together with fishbone diagrams aimed at knowledge generation and analysis. Fishbone diagrams are similarly known as cause and effect diagrams, and are a portion of well-known business problem and analysis methods announced by Cross [16]. The modest queries develop an agile process that works perfectly for the team and facilitates it to succeed amongst the complexity of its environment. By "appropriately," we mean that the Retrospective facilitates choice, conservation, and innovation not objective for one Sprint, but frequently across several "generations" of the Team:

- Ask "what went well." The Scrum Master should enquire the Team what it wants to recurrence such virtuous effects; acquire recognized licenses for the tools that were used on a trial basis and worked well; share prosperous behavior with other teams in short, effort to support these good team traits.
- Ask "what didn't go well," as an approach to choice out traits that offended the Team, and for every "didn't go well" item, be sure to enquire:
- "What could we change to improve this issue?" For every "didn't go well" item, the Team originates to recognize activities it can own, or that the organization necessity to possess. Team-owned variations moreover converted Backlog Items, or behaviors of which the Scrum Master necessity to retain the team aware. Organization-owned variations become Organizational Impediments which the Scrum Master essential work to resolve.

  The succeeding points to involve team participants and conduct a productive retrospective.

- Review the consequences from earlier retrospectives against enhancements in the present sprint.
- Gain the sponsorship of somebody who has the supremacy to do somewhat with the consequences of the retrospective.
- Create interval for face-to-face deliberations with all members before the retrospective.
- Enquire every person, "What might prevent you from presence fully present during the retrospective?"
- Be enthusiastic to variety modification, such as reorganizing or shifting more essentials of the design, to confirm the utmost level of contribution.
- Review the retrospectives goals and schedule with every person.
- Enquire every member for recommendations about what would create the retrospective even healthier for them.
- Repeat every person that if the team desires things to variation it necessity to demonstration the statistics, its meaning and significance in behaviors that will compel the sponsor to respond.
- Enquire every person "Is there anything that I should have enquired you that I didn't?"
- Express every person in what way significant their contribution is to you personally.
- Receiving the team participants to a retrospective by force would not solve the purpose. This would create them physically existing nevertheless they would still be conceptually absent-minded.

Therefore, in order to make a retrospective effective, the team wants to be present through satisfactory groundwork and active involvement. The team and stakeholders must be organized to yield corrective achievement at the finish of every retrospective.

In this paper, we will deliberate practical approaches to harvest knowledge after projects that are either finish of sprint a major activity or phase. We refer to these approaches as 'Retrospective reviews' as this is a common term. The key objective of this artifact is to focus the significance of group-processes as an approach for knowledge distribution in software projects, and to provide a summary of identified such practices in the arena of software engineering. We organized as follows: section 2 encourage this investigation in more features and provide background on associated work in the arena of Agile Retrospective review. Section 3 case study analyses on retrospective approach, used in Agile software development, Section 4 presents the results and assesses the case study on knowledge acquired from projects that are either completed or have finished every sprint iteration, Section 5 summarizes future research directions that we identified based on the case study and concludes the paper.

RETROSPECTIVES REVIEW

Around numerous concepts of in what way wisdom takes place in conventional effort circumstances. Brown and Dugid [17] have described in what way "societies of practice" acquire from each other, David Kolb [18] describes in what way persons acquire from experience, and Nonaka and Takeuchi [7] have established a "theory of knowledge construction". Learning also takes place in software development. New skills, work approaches as well as problems in existing software development approaches need that software developers are keen on learning.

Learning Knowledge Model

Towards deliberate learning in software engineering, we will trust on the model presented by Nonaka and Takeuchi [7]. We select this model, because of its emphasis on tacit knowledge, which shows how learning takes place in agile development methods. It is also a model that is normally recognized. They divide amongst two forms of knowledge: Tacit knowledge that persons have, but are incapable to characterize. Other kind of knowledge is explicit knowledge – which is thinkable to characterize as process strategies or several other form of documentation.

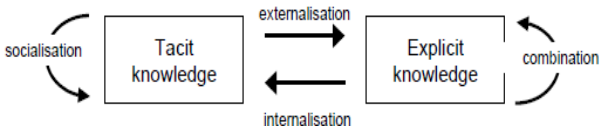

Figure 1: Learning Knowledge Model of Tacit and Explicit Knowledge

Further, Nonaka and Takeuchi [7] distribute four approaches of knowledge conversion amongst tacit and explicit knowledge, as shown in Figure 1:

- Socialization.
- Internalization.
- Externalization.
- Combination.

According to Nonaka and Takeuchi [7] knowledge passes through diverse ways of conversion in an agile which creates the knowledge extra distinguished, and similarly spreads it across diverse layers in an organization.

Learning Agile

Extreme programming (XP) accentuates on teamwork and communication within software development teams. The innovator of XP, Kent Beck, suggests that the bottlenecks in software development entirely team communication amongst persons is struggle [19]. One of the 13 practices in XP [3], pair programming, is deliberate to confirm that the developer acquire from each other. Pair programming is a practice that maintenances transfer of tacit knowledge, as Barry Boehm opinion in an artifact, "agile approaches originate considerably of their agility by trusting on the tacit knowledge personified in the team, rather than writing the knowledge down in plans" [20]. Pair programming mechanisms as knowledge transfer of whatever is called socialization in Nonakas model. No knowledge is documented, except in the source code, means that considerable significant knowledge is inaccessible to others than the ones that work in each project.

The impression of the retrospective, well-defined in agile principles is that, it is a space for teams to examine and adapt, to acquire around what works and what does not work, and to discover healthier techniques of working organized and through their product owner, forever motivated towards the lean principle as shown in the Figure 2. Aimed at to occur the team need retain these key aims in mind:

- Open and Truthful Communication
- Team Thoughtful
- Executable Action Articles

Without Executable Action Articles you are merely clearing the air, either griping or congratulating without point or determination. This artifact is the ultimate harvest of the retrospective meeting; a sanctioned strategy on what fundamentals of a team's working situation or practices to variation in order to progress performance over past sprint.

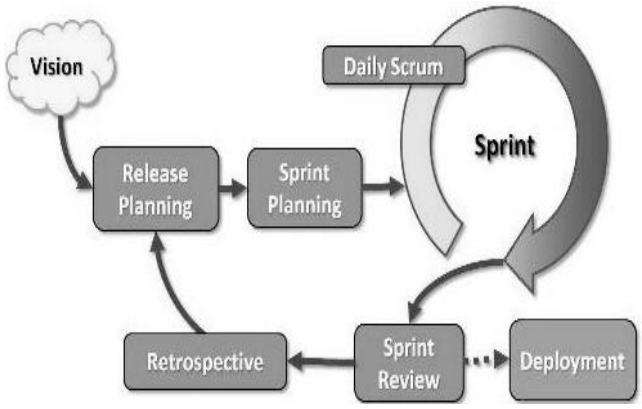

Figure 2: Agile Retrospective Review

The Executable Action Articles are a very small check list or to-do list of recommended thoughts, also recognized as experiments, for enlightening the performance of the team. The lists of open concerns are scoped down to a small list of high significance objects, which the team desires to discourse in the succeeding sprints. We request these Executable Action Articles since every article would need certain action by the team, must be executable, quantifiable and evident. Occasionally this is as modest by way of adapting the existing team arrangements. Occasionally this needs a variation in practices. Occasionally this needs innovative designs of collaboration through outside sources such as the Product Owner, or outside teams similar IT, or Release Management. All retrospective must harvest one or more perfect Executable Action Articles that remain found from the team, decided upon by the team, and utmost of all dedicated to implementation by the team. This article must be documented for historical situation or further mutual resource.

Learning Retrospective

Retrospective means a shared learning action which can be structured for projects moreover after they finish of a sprint or terminated. The key inspiration is to reproduce on what occurred in the project in order to progress forthcoming practice for the persons that must contributed in the project and aimed at the business as a entire. The physical consequence of a conference is a Retrospective report. This category of practices has also remained referred to as 'autopsy review', 'project analysis review', 'project retrospectives', 'Santayana review', 'Retrospective analysis', 'post project review', 'after action reviews', 'quality improvement review' and 'touch-down meetings'. Researchers in organizational wisdom occasionally practice the word 'reflective practice', which can be well-defined as 'the practice of intermittently stepping back to consider on the significance to identity and others in one's instantaneous situation around what has freshly emerged. It illuminates whatsoever has remained knowledgeable through both self and others, providing a foundation aimed at forthcoming action' [21]. This includes uncovering and creating clear outcomes of scheduling, opinion and accomplished practice. It can lead to thoughtful of experiences that must remain unnoticed in practice. In the model of Nonaka and Takeuchi, Retrospectives continue a combination of knowledge through socialization and through externalization. In listening towards others you service socialization and in reproducing and distribution your own practice you convey your tacit knowledge. Retrospectives remain likewise a technique for leveraging knowledge after the individual level to the organizational level. In a study on vital practices in research and development-companies, 'learning from retrospective audits' remain understood as unique of the utmost talented practices that might harvest reasonable benefit [22]. A study on retrospective reviews in research and development businesses demonstration that only one out of five projects expected a retrospective review [23]. Furthermore, the reviews incline to motivation on practical harvest and bureaucratic quantities. Kransdorff [24] analyzes Retrospectives because persons contributing do not have an accurate remembrance, which can lead to arguments. Kransdorff recommends gathering data through the project, with small discussions, in a determination to become extra objective material.

Retrospective Reviews in Agile Software Development

There are numerous methods to implement ARR. Apple company has used a approaches [14] which comprises designing a project study, accumulating objective development material, conducting a debriefing seminar, a 'development history day' and lastly distributing the consequences. At Microsoft they similarly put considerable work into writing 'Retrospective reports'. These comprise conversation on 'what worked well in the last sprint, what did not work well, and what the group must do to progress in the next sprint [25]. The size of the resulting document is quite large. Watts Humphrey recommends an approach to do Retrospectives to 'acquire what went accurate and erroneous, then to realize in what way to do the business healthier the next sprint [26]. A explanation of another lightweight method which seeks to provoke experience by meetings, and not a group process, is designated by Schneider [27]. Norman Kerth lists a entire of 19 methods towards be used in Retrospectives [28], certain concentrating on making an atmosphere for conversation in the project, certain for studying the previous development, certain for assisting a team recognize and embrace modification throughout their succeeding development, and certain for distributing with the unique belongings of a unsuccessful development. Tiedeman [29] recommends three categories of Retrospectives, associated to an agile software development; one aimed at 'planning development', one aimed at 'design work' and one aimed at 'Retrospective review' towards afford criticism afterwards the established system has remained in practice for certain time.

Methods on ARR

We have nominated approaches that can be implemented in small time, and remain thus applicable even aimed at small and medium-size companies. Neal Whitten recommends the succeeding practice aimed at conducting retrospective reviews [30]:

- Declare resolved.
- Choice contributors.
- Organize aimed at workshop.
- Conduct workshop.
- Present consequences.
- Adopt references.

Collison and Parcell [31] recommend the succeeding phases aimed at establishing a retrospect seminar:

- Call the seminar.
- Invitation the accurate persons.
- Assign a facilitator.
- Reexamine the aims and deliverables of the development.
- Reexamine the development proposal or method.
- Request 'What went well'.
- Why these characteristics went well, and express the knowledge as assistance aimed at the forthcoming.
- Request "what might have gone healthier".
- Discovery available what the problems.
- Confirm that the contributors leave the seminar through their approaches recognized.
- 'What next'.
- Recording the seminar.

Birk et al. have used Retrospective Reviews as a collection practice [14, 29, 32, 33, 34], where utmost of the effort is finished in single seminar eternal half a day. The Retrospective seminar has succeeding phases:

- Summary. Presented the program of the day and the determination of the Retrospective review.
- **KJ Seminar 1.** Hand out post-it notes and request team member to write down whatever went well in the development, receive demonstrations, assembly the concerns on the whiteboard, and provide them significances.
- **KJ Seminar 2.** Hand out post-it notes and request team member to write down issue that looked in the development, receive demonstrations, assembly the concerns on the whiteboard, and provide them significances.

CASE STUDY ON ARR

We have realized diverse methodologies to leading retrospective reviews. First, we present the company, before the development on where the review remained carried out, then case study and methodologies and finally extracts from the retrospective report. The instance reported now remained selected since of an extensive data gathering by way of a part of an action research [35] development on software process improvement.

Supply Chain Management (Scm) Software Company

The company creates software aimed at manufacturer and a wholesaler, or between a wholesaler and a retailer. They are distributed SCM systems, consultancy services, feasibility studies, system engineering, training, and support. The company has remained working with large development projects; both as a prime contractor and as a subcontractor are working. The companies possess a steady and extremely accomplished staff, several with master's degrees in Computer Science and have an 'engineering culture'. Approximately 3000 people remain working in the company, and the majority is working with software development. Development projects are accomplished in accordance with SCM standards, and are usually fixed price projects.

SCM can undertake accountability for a whole complex, multi-national SCM program or handle a smaller subset of selected projects, areas or trading partner relationships. More than 250 customers, including many from the Fortune 500 from around the world, presently trust in SCM Skillful Operations. The company had difficulties with estimating the size of new software projects. Several people in the company also felt that they did not handover sufficient experience amongst their software development projects. Project wrote an 'experience report', but these were infrequently measured interesting, and remained not delivered very often. To progress this, the company decided to attempt retrospective reviews at the end of sprint iterations.

Case Study Approach

We systematized a retrospective in SCM software project. We decided to emphasis on some of the positive values that create up a virtuous team: *Simplicity, Feedback, Courage, Respect and Communication*. We attempt to acquire as several as thinkable of the persons who have been working in the development to contribute in retrospective review. The objective of this seminar is to accumulate information from the contributors, create them deliberate the approach the development was carried out, and similarly to examine reasons for why things worked out well or did not work out. The 'requirements' aimed at this practice is that it must not take considerable time for the development team to contribute, and it must deliver a opportunity for deliberating utmost significant knowledge from the development, composed through an study of this knowledge. The key outcomes are recognized in a report and entirely contributors in a development are requested to retrospective seminar. Certain of the approaches we required to attempt we hadn't used earlier in our team. Our sprints or iterations are Bi-Weekly, so we discovery that a Two hours retrospective is sufficient time to deliberate and effort through the problems of the Bi-week sprints or iteration and the Retrospective meeting has succeeding phases or agenda.

Agenda

- Check in - 15 mins
- Reading Prime directive - 15 mins
- Brainstorm Values - 15 mins

- Dot voting Values - 30 mins
- Deliberate consequences - 30 mins
- Retrospective feedback - 15 mins

Total: 2 hours

The entire quantity of resources desired for a retrospective review were designated entirely the development contributors and Facilitators. The entire effort required was 24 person hours, distributed by way of surveys:

- 12 members in a team containing Facilitators
- 2 person hours for the retrospective review meeting on each sprint cycle.
- Overall effort desirable was 24 person hours for the retrospective review meeting on each sprint cycle.

Check in

The retrospective kicks off through a brief check in. It's a chance aimed at each team participant to rapidly state in what way they felt around the previous sprint. Portion of its significance is repeating every team member that their opinion is significant. Through a team of 12 people, certain added vocal than others, it's significant towards esteem everyone's judgment.

Aimed at check in, every participant will designate in what way they felt around the previous sprint, nevertheless we diverged somewhat from the normal organization through enquiring that they limit their response to simply one of five adjectives:

(1) Happy, (2) Angry, (3) Apprehensive, (4) Sad, (5) Hopeful

Their responses were mostly “Happy” through a little “Hopeful” and single “Angry”; we resolve deliberate more on that angry later.

Prime Directive

Subsequently check in was completed, team participant towards volunteer to deliver available the prime directive. Retrospectives are around learning, improving and reproduction. Participant can be vocal in retrospectives; nevertheless belongings must continue confident and not incline into destructive blame and personal criticism. The arguments of the prime directive repeat everybody that: ”Nevertheless of whatever we learn, we recognize and accurately trust that everybody did the best work they might, given whatever they recognized at the time, their skills and aptitudes, the resources accessible, and the condition at hand”. Formerly, through a demonstration of thumbs, the team elected on if they agree through the statement. The majority of participant usually votes thumbs up. Occasionally participant has a half approach vote, this inclines to be a personal criticism of their own performance, rather than absorbed towards their team mates.

Brainstorm Values

We prepared up five notices on the wall of the meeting area by way of *Simplicity, Feedback, Courage, Respect and Communication*, as demonstration in the Figure 3. The team spent conversation around and essential what each of the values mean in relative to the team. Aimed at a motivated brainstorm on anything occurred in the development, a method named after a Japanese ethnologist, Jiro Kawakita [8] - called ‘the KJ Method’ is used aimed at innovation.

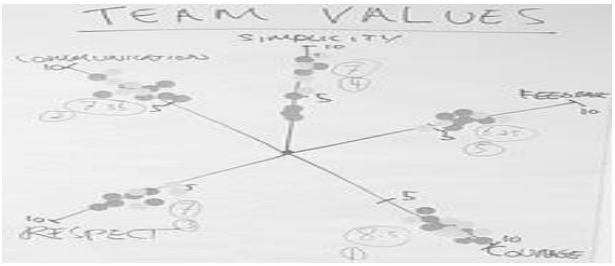

Figure 3: Brainstorm Value and Dot Voting

Dot Voting

Every team participant to vote on a scale of one to ten on each of the Principles, through a one means we remained not honoring the value and a 10 exceeding in the Value. Ten notes are supplied out to every participant in a team and requested the participants to assign one note to a whiteboard and approximately why this remained significant. Formerly the succeeding participant presents a note and so on until all the notes are on the whiteboard. The team positioned notes on a star chart. The usual rating of each value was agreed on as shown in Figure 3. The notes are formerly grouped, and every group is specified a new name. The team fingered Courage was our strength, followed on by Communication, then Respect, Simplicity and lastly Feedback.

DISCUSSIONS AND RESULTS

We followed up the consequences through requesting why the team understood feedback was voted as one of our weakest values. Root Cause Analysis similarly called Ishikawa or fishbone diagrams are used to examine the reasons of significant concerns. The facilitator draws an arrow on a whiteboard representing the concern being deliberated, and assigns other arrows to this one similar in a fishbone through concerns the participants deliberate are producing the first concern. Occasionally, similarly essential aims for certain of the key reasons are committed as well. We transcribed a retrospective report on the development, comprising a summary which designated the practice, a tiny explanation of the development that we investigated in what way the investigation was carried out, and the consequences of the investigation. The consequence was an ordered list of difficulties and achievements in the development. We used statements from the meeting to present whatever was thought around the concerns through utmost precedence, composed through an Ishikawa diagram to demonstration their root causes. We comprised everything that was transcribed down on notes during the KJ meeting of Figure. 4, and a transcript of the demonstration of the concerns that were used on the notes.

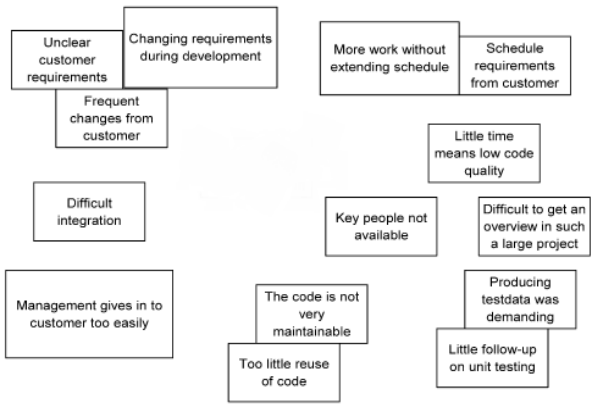

Figure 4: Notes in Agile Software Development Project after A KJ Session

Single consequence from the KJ session on difficulties that looked in the development was notes collected together and titled ‘changing requirements’ as shown in the Figure 4. Once we carried this concern up over in order to

discovery certain of the root causes for 'changing requirements', we finished up through the fishbone diagram as shown in the Figure 5. The root causes for the varying requirements, as the participating in the investigation saw it, was that the requirements were poorly specified by the customer, there were 'new requirements' throughout the development and the requirements are varying frequently during mid of sprints and requesting criticism within short span of duration and the company recognized tiny of whatever the customer was doing. An action was established to report this. This was the source of the anger mentioned at the "check in" section. We wanted to finish the deliberations on the confident, so we requested a similar query around courage and why that was voted team strength. We deliberate that the extensive spread of votes concerning simplicity was value discussing.

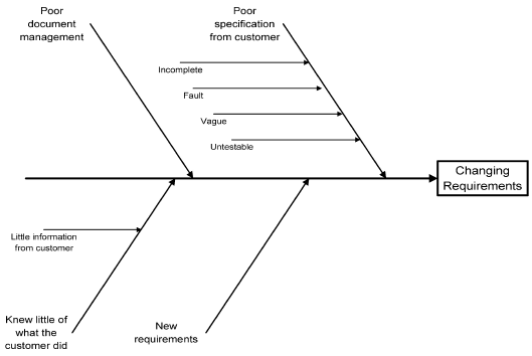

Figure 5: Fishbone Diagram

Retrospective Feedback

We closed the retrospective through return on time invested chart (ROTI). Agile is around self-analysis and constant improvement. It's significant to apply the similar values to the retrospectives too. We desired to employ certain minute receiving feedback in what way the retrospective can be enhanced succeeding time. This kind of retrospective established 4 votes saying that it was a received benefit equal to time invested and 8 votes saying that it was a received benefit greater than time invested out of 12 participate as shown in the Figure 6.

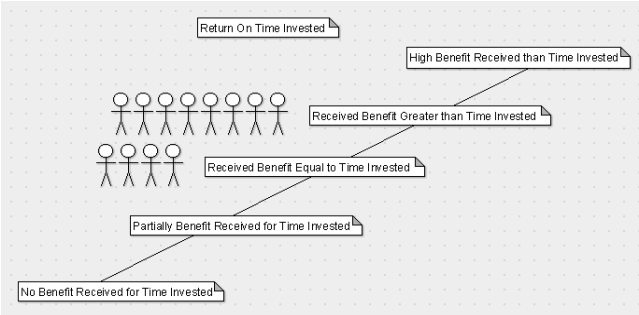

Figure 6: Return on Time Invested Chart

We used a recording section throughout the demonstrations, and recorded everything that is said. We transcribed a retrospective report around the development, which comprise a summary, a tiny explanation of the development investigated, in what way the investigation was carried out, and the consequences of the investigation. The consequence is a prioritized list of difficulties and achievements in the development. Statements from the meeting are used to present whatever was said around the concerns with utmost priority, composed with a fishbone diagram to demonstration their causes. All that was transcribed down on post-it notes throughout the KJ session is comprised, as well as a recorded of the demonstration of the concerns that were used on the post-it notes. Such reports are typically amongst 2 to 5 pages in length. Afterwards the meeting, we offered the report to participates elaborate in the development to collect feedback and organize

slight improvements. Afterwards the retrospective meeting was completed; we requested participate to state whatever they understood of the practice. All participants had got new visions on the development were talented to see concerns from new viewpoints. Also, several specified that the approach of conducting retrospective was motivating in itself because it was unusual.

CASE STUDY CONSEQUENCES

Keegan and Turner [38] establish that 'development team participants often do not have the time aimed at meetings, or for sessions to evaluation lessons learned. Keegan and Turner do not deliberate what kind of retrospective practices occurred in the companies, nevertheless the key outcome was that the process were occasionally used in practice.

We deliberate there is a essential aimed at facilitating software companies selecting modest and practical approaches for showing retrospectives, to create it easier to accomplish retrospectives to a higher grade. The assistances of showing retrospective reviews are primarily that it delivers a learning opportunity wherever deliberations are appropriate to the development and to the company. It can also be an approach for management to demonstration that they attend to what the employees say, and are keen to deliberate progress efforts.

Effective Retrospective Process

Retrospective delivers team participants a chance to discover what is working well & what needs to be improved. It not only permits team to create change for the forthcoming, nevertheless also authorize that virtuous practices are reinforced and recurring. To create the retrospective more effective succeeding are suggested phases which can assistance creating an excessive team with the assistance of retrospective.

- Strategy and carry on aimed at all sprints / [2-3 weeks]. Healthier to carry on at the finish of sprint earlier the succeeding sprint initiates.
- Organize it through a lesser team (Less than 20 people].
- Make certain entirely participants have the corresponding skills and are dedicated to a mutual objective.
- Request every individual to come prepared to the retrospective meeting.
- Establish a harmless situation somewhere team can expose delicate areas and at the similar time have an expressive discussion. Team must be opening sufficient to deliver contributions.
- Established ground guidelines similar & confirm equal participation from all attendees.
- In a retrospective this is wanted by having a skilled practice leader who boosts open dialogue, and must prevent critique of individuals and that dominating people get the utmost of the meeting time.
- Analysis your metrics and use it as an approach to must a good meaningful conversation (e.g. defects from earlier sprint, team temperature etc.)
- Capture entirely concerns, nevertheless prioritize & brainstorm on only highest 2-3 concerns which essential rapid actions. Take action for next sprint rather taking extended stretch goal
- Norman Kerth [28] recommends the significance of a good atmosphere through the 'prime directive': "Regardless of whatever we determine, we need to recognize and really trust that everybody did the greatest job

he or she might, specified whatever was recognized at the time, his or her talents and capabilities, the resources offered, and the condition at hand".

- Preserve a backlog to retain additional recommended enhancements.
- Analysis action objects from preceding retrospective and yield remedial actions.
- Openness, patience, the capability to listen, investigation through innovative words and ideas, politeness, the creation of a believable argument and bravery are certain components aimed at a virtuous conversation [36].
- It is hard to provide assistance on whatever is the 'optimal' interval practice aimed at a retrospective. Additional interval will permit more concerns to be deliberated deeper, therefore accumulative the learning effect. The interval used for a retrospective must depend on what kind of approach a company has categorization needs additional effort than personalization. It must also depend on the scope of the development, as there must be additional concerns to deliberate in a larger development than in a small.

Organizer

Whitten, Collison, Parcell and Birk et al. recommends using a facilitator aimed at the meeting. The query is what thoughtful of participate is the accurate to use. Perhaps intelligent to use somebody from external of the development, whom the participants trust. It can similarly be somebody who is outside to the company. An advantage of using an outside person is that participants must to describe concerns to this person more systematically than they would to an insider. This can reason diverse understandings inside the development to be uncovered. The facilitator must also be accurately skilled in order to continuation when declarations from persons are unclear.

Guidelines for Retrospective Organizers are as follows:

- When you enquire a query to the team, certain sufficient time to response for participate.
- Do certain of the effort in pairs or small team.
- Let the team do as much of the effort as potential.
- Provide directions in chunks.
- Offer a significant when you color code whatever as portion of an action.
- If you get vanished in an action or somewhat happens that you don't recognize fairly in what way to handle, pause and reset.
- Usage extensive chisel tip markers in dark colors for lettering on flip plans or white boards.
- Transcribe big, and use sentence case when lettering on flip plans or white boards.

Spectators

The approaches designated entirely argue that one must offer an extensive audience aimed at a retrospective. Whitten recommended participants from planning, development; test, usability and module build as illustration characters to offer. Collison and Parcell recommend that participants from a similar development that are proceeding must be requested as well as essential participants from the development. The approach of Birk et al. recommends receiving "as numerous participants as imaginable" from the development to participate. Observing back at the approaches for awareness

distribution, it appears reasonable that entirely participants in a development can contribute with knowledge that is applicable for forthcoming developments over socialization.

Management

Must the management or the development supervisor take portion throughout a retrospective? We do not deliberate the management must take portion in the retrospective, as the objective is to emphasis on learning, and organization also has a role of gauging participants. This can be a problematic as we saying in the software development example, where organization was blamed aimed at certain of the difficulties in the development. Nevertheless this kind of difficulties can be deliberated with organization afterwards the retrospective meeting is completed. The development supervisor is very valuable to comprise because this person has a more complete opinion of the development than the rest of the participants. Nevertheless this person can also be fast to protect entirely choices occupied throughout the development, and create it problematic to have a permitted discussion of thoughts on in what way to progress the next development. Assumed a strong facilitator that is conscious of the conceivable difficulties with the development supervisor, we deliberate a development supervisor must be requested to become a more comprehensive summary in the retrospective.

Retrospective Consequence

Whitten designates a list of recommendations that are specified to the company management in order to guarantee learning in other developments. Collison and Parcell similarly recommend such strategies aimed at the forthcoming, nevertheless mention histories to demonstrate the strategies, designations of people involved and fundamental artifacts. Birk et al. recommends writing a report which designates the development, what went well, what went wrong, and the causes of what went well and wrong. They also transcribe considerable of what is supposed throughout the meeting in order to stretch extra situations for forthcoming readers. If the objective of the retrospective primarily is too derived with improvement recommendations, possibly the approaches designated by Whitten is enough. Nevertheless if the purpose is to handover knowledge also to participants who did not take part in the retrospective, the method of Birk et al. is more suitable. Kerth [28] recommend that the participants in the retrospective meeting must transcribe the report; otherwise they lose guarantee to the comfortable. The Cross Affinity Exercise [28] yields suggestions for variation, which recognizes participants enthusiastic to work on the variation.

CONCLUSIONS AND FUTURE WORK

In this paper, we have introduced a new approach to the process of agile retrospective analysis of software development projects. According to our initial experience, the results of the project reviews in small projects are also promising. The chosen ARR practice and ideas were found to work well and especially in agile software development projects with tight schedules. Competence building and the creation of further knowledge is needed, if retrospective analysis is to be used systematically in experience and project-based learning. We have investigated retrospective reviews from a knowledge management perspective and presented methods for conducting agile retrospective from the literature. The methods vary in several dimensions. They put different emphasis on who to invite, how to prepare, how to facilitate the retrospective meeting, how to structure discussions, and what the written output of the retrospective is to be. Companies wanting to conduct agile retrospective should decide on the method to use after what general strategy they have for knowledge management. They should also decide whether they want to focus purely on internal project affairs, or also to include relations to project stakeholders.

REFERENCES

1. T. Dyba°, Enabling Software Process Improvement: An Investigation on the Importance of Organisational Issues, Dr. Ing. Thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, 2001.
2. S.W. Ambler and M.J. Lines, Disciplined Agile Delivery: A Practitioner's Guide to Agile Solution Delivery in the Enterprise. IBM Press, 2012.
3. K. Beck, "Embracing Change with Extreme Programming," IEEE Computer, pp. 70 - 77, October, 1999.
4. A. Cockburn, Agile Software Development. Boston: Addison-Wesley, 2002.
5. T. Dingsøyr and G. K. Hanssen, "Extending Agile Methods: Postmortem Reviews as Extended Feedback," presented at 4th International Workshop on Learning Software Organizations (LSO'02)), Chicago, Illinois, USA, 2002.
6. K. Beck, Extreme Programming Explained: Embrace Change: Addison Wesley Longman, Inc., 2000.
7. I. Nonaka and H. Takeuchi, The Knowledge-Creating Company, 1995.
8. R. Scupin, "The KJ Method: A Technique for Analyzing Data Derived from Japanese Ethnology," Human Organization, vol. 56, pp. 233-237, 1997.
9. B. Collier, T. DeMarco, and P. Fearey, "A defined process for project post mortem review," IEEE Software, vol. 13, pp. 65-72, 1996.
10. Terry Williams : Identifying the hard lessons from projects – easily; International Journal of Project Management, Volume 22, Issue 4, pp. 273-279, May 2004
11. Tor Stålhane, Torgeir Dingsøyr, Geir Kjetil Hanssen, and Nils Brede Moe: Post Mortem – An Assessment of Two Approaches; In Reidar Conradi and Alf Inge Wang (Eds.). Empirical Methods and Studies in Software Engineering: Experiences from ESERNET, LNCS 2765. Springer Verlag, pp. 129-141, 2003
12. S.A. Ambler, "How agile are you? 2010 survey results," 2010.

    http://www.ambysoft.com/surveys/howAgileAreYou2010.html
13. Torgeir Dingsøyr, Tor Stålhane and Nils Brede Moe: A practical guide to Lightweight Post Mortem Reviews. University of Oslo. [For teaching]; http://www.uio.no/studier/emner/matnat/ifi/INF3120/h03/undervisningsmateriale/Prosjektoppgave/PMA_practical_guide.pdf. [Retrieved March 30th 2004], 2003
14. [1s] Andreas Birk, Torgeir Dingsøyr, and Tor Stålhane: Postmortem: Never leave a project without it; IEEE Software, Special Issue on Knowledge Management in Softwa re Engineering, pp. 43-45, May-June 2002
15. John S. Reel: Critical Success Factors in Software Projects; IEEE Software, pp. 18-23, May/June 1999
16. Anthony T. Cross: Management Through Visibility: Business Improvement; Murton Group, 2003
17. J. S. Brown and P. Duguid, The Social Life of Information. Boston, Massachusets: Harvard Business School Press, 2000.
18. M. Kennaley, SDLC 3.0, Beyond a Tacit Understanding of Agile. Fourth Medium Press, 2010.

19. K. Beck, “Extreme Programming: A Humanistic Discipline in Software Development,” presented at Fundamental Approaches to Software Engineering. First International Conference (FASE'98), 1998.
20. B. Boehm, “Get Ready for Agile Methods, with Care,” IEEE Computer, pp. 64 - 69, January, 2002.
21. B. Boehm, “Get Ready for Agile Methods, with Care,” IEEE Computer, pp. 64 - 69, January, 2002.
22. A.W. Brown, “A case study in agility-at-scale delivery,” Proc 12th International Conference on Agile Software Development, XP2011, Springer Verlag, May 2011.
23. M. Zedtwitz, Organizational learning through post-project reviews in R&D, R&D Management 32 (2002) 255–268.
24. A.W. Brown, Global Software Delivery: Bringing Efficiency and Agility to the Enterprise. Addison-Wesley, 2012.
25. J. Sutherland, “Ten year agile retrospective: how can agile improve in the next ten years?” 2011.
26. W.S. Humphrey, The postmortem, in: Introduction to the Team Software Process, Addison Wesley Longman, Reading, MA, 1999, pp. 185–196.
27. K. Schneider, LIDs: a light-weight approach to experience elicitation and reuse, Second International Conference on Product Focused Software Process Improvement, PROFES 2000, 2000, pp. 407–424.
28. N.L. Kerth, Project retrospectives: a handbook for team reviews, Dorset House Publishing, New York, 2001.
29. P. Kruchten, “Contextualizing agile software development,” Journal of Software Maintenance Evolution-R, 2011
30. D. Leffingwell, Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Pearson Education, Inc., 2011.
31. C. Collison, G. Parcell, Learning to Fly: Practical Lessons from one of the World’s Leading Knowledge Companies, Capstone Publication, 2001.
32. T. Dingsøyr, N.B. Moe, Ø. Nytrø, Augmenting experience reports with lightweight postmortem reviews in: F. Bomarius, S. Komi-Sirvio¨ (Eds.), Third International Conference on Product Focused Software Process Improvement, Springer, Kaiserslautern, Germany, 2001, pp. 167–181.
33. T. Sta°lhane, T. Dingsøyr, N.B. Moe, G.K. Hanssen, Post mortem—an assessment of two approaches, EuroSPI, 2001, pp.
34. T. Dyba°, T. Dingsøyr, N.B. Moe, Process Improvement in Practice —a Handbook for it Companies, Kluwer, Boston, 2004.
35. P. Kruchten, “Contextualizing agile software development,” Journal of Software Maintenance Evolution-R, 2011.
36. G.v. Krogh, K. Ichijo, I. Nonaka, Enabling Knowledge Creation, Oxford University Press, New York, 2000.